(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2000－259417
（P2000－259417A）
(43)公開日　平成12年９月22日(2000.9.22)

(51)Int.Cl.[7]	識別記号	ＦＩ	ﾃｰﾏｺｰﾄﾞ(参考)
Ｇ０６Ｆ　9/445		Ｇ０６Ｆ　9/06	４２０Ｃ　５Ｂ０７６

審査請求　未請求　請求項の数30　ＯＬ　（全 21 頁）

(21)出願番号　特願平11－67028

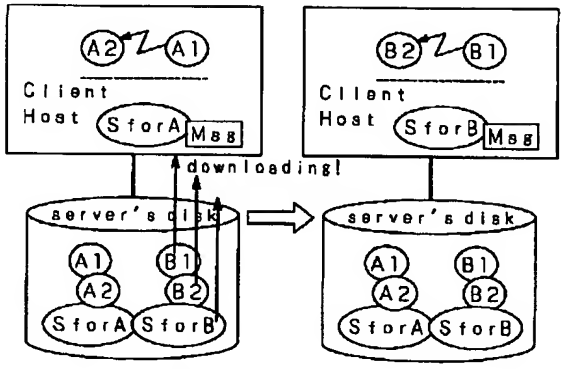
(22)出願日　平成11年３月12日(1999.3.12)

(71)出願人　000002185
ソニー株式会社
東京都品川区北品川６丁目７番35号
(72)発明者　岡村　英明
東京都品川区北品川６丁目７番35号　ソニー株式会社内
(74)代理人　100067736
弁理士　小池　晃　（外２名）

Ｆターム(参考)　5B076 BB06 BB14 BB16 BB19 EA03

(54)【発明の名称】　データ処理装置、データ処理方法及びプログラム提供媒体

(57)【要約】
【課題】　オブジェクト指向を適用したデータ処理システムにおいて、最適な実行環境が異なる複数の実行主体（アプリケーションプログラム等）があるときに、各実行主体に対してそれぞれ好適な実行環境を提供できるようにする。
【解決手段】　１つ以上のオブジェクトから構成される少なくとも１つの実行主体と、上記実行主体に対してサービスを提供する１つ以上のオブジェクトから構成される少なくとも１つの実行環境とを備えたデータ処理システムにおいて、実行主体又は実行環境を構成するオブジェクトからの要求に基づいて、実行環境を構成するオブジェクトを削除する削除ステップと、外部システムから新たなオブジェクトを取り込む取込ステップとを行い、実行環境を構成するオブジェクトを置換する。



システムオブジェクトダウンロード

1

【特許請求の範囲】

【請求項１】　１つ以上のオブジェクトから構成される少なくとも１つの実行主体と、

上記実行主体に対してサービスを提供する１つ以上のオブジェクトから構成される少なくとも１つの実行環境と、

上記実行主体又は実行環境を構成するオブジェクトからの要求に基づいて、上記実行環境を構成するオブジェクトを削除するとともに外部システムから新たなオブジェクトを取り込んで、上記実行環境を構成するオブジェクトの置換処理を行うオブジェクト置換手段とを備えることを特徴とするデータ処理装置。

【請求項２】　上記オブジェクト置換手段は、上記実行環境を構成するオブジェクトからなることを特徴とする請求項１記載のデータ処理装置。

【請求項３】　上記実行環境を構成するオブジェクトは、当該オブジェクトの優先度を示すダウンロード受諾レベルを備え、

上記実行主体を構成するオブジェクトは、当該オブジェクトの優先度を示すダウンロード許可レベルを備えることを特徴とする請求項１記載のデータ処理装置。

【請求項４】　上記オブジェクト置換手段は、上記実行主体又は実行環境を構成するオブジェクトからオブジェクト削除要求を受けとった場合、オブジェクトの削除を要求したオブジェクトが備えるダウンロード許可レベルと、削除対象オブジェクトが備えるダウンロード受諾レベルとを比較し、その比較結果に基づいてオブジェクト削除の可否を判定し、オブジェクト削除可と判定された場合にだけ、オブジェクトの削除処理を行うことを特徴とする請求項３記載のデータ処理装置。

【請求項５】　上記ダウンロード受諾レベル及び上記ダウンロード許可レベルには、オブジェクトの優先度に対応した数値が設定され、

上記オブジェクト置換手段は、オブジェクトの削除を要求したオブジェクトが備えるダウンロード許可レベルが、削除対象オブジェクトが備えるダウンロード受諾レベルよりも大きい場合に、オブジェクト削除可と判定することを特徴とする請求項４記載のデータ処理装置。

【請求項６】　上記オブジェクト置換手段は、上記実行主体又は実行環境を構成するオブジェクトからオブジェクト取込要求を受けとった場合、オブジェクトの取り込みを要求したオブジェクトが備えるダウンロード許可レベルと、取込対象オブジェクトが備えるダウンロード受諾レベルとを比較し、その比較結果に基づいてオブジェクトの取り込みの可否を判定し、オブジェクト取込可と判定された場合にだけ、外部システムからのオブジェクトの取り込みの処理を行うことを特徴とする請求項３記載のデータ処理装置。

【請求項７】　上記ダウンロード受諾レベル及び上記ダウンロード許可レベルには、オブジェクトの優先度に対応した数値が設定され、

2

上記オブジェクト置換手段は、オブジェクトの取り込みを要求したオブジェクトが備えるダウンロード許可レベルが、取込対象オブジェクトが備えるダウンロード受諾レベルよりも大きい場合に、オブジェクト取込可と判定することを特徴とする請求項６記載のデータ処理装置。

【請求項８】　上記実行環境を構成するオブジェクトは、当該オブジェクトの優先度を示すダウンロード受諾レベルを備え、

上記実行主体を構成するオブジェクトは、当該オブジェクトの優先度を示すダウンロード許可レベルを備え、

上記オブジェクト置換手段は、上記実行主体を構成するオブジェクトからの要求に基づいてオブジェクトの置換処理を行う際に、オブジェクトの置換を要求したオブジェクトのダウンロード許可レベルと、置換対象オブジェクトのダウンロード受諾レベルとを比較し、その比較結果に基づいてオブジェクト置換の可否を判定し、オブジェクト置換可と判定された場合にだけ、オブジェクトの置換処理を行うことを特徴とする請求項１記載のデータ処理装置。

【請求項９】　上記実行環境を構成するオブジェクトは、サービスを提供する対象のオブジェクトが登録されたサービスリストを備え、

上記オブジェクト置換手段は、オブジェクトの置換処理を行う際に、削除対象のオブジェクトのサービスリストに登録されたオブジェクトの状態を検査し、その検査結果に基づいてオブジェクト置換の可否を判定し、オブジェクト置換可と判定された場合にだけ、当該サービスリストの内容を一時記憶領域にコピーした上でオブジェクトの置換処理を行うことを特徴とする請求項１記載のデータ処理装置。

【請求項１０】　上記オブジェクト置換手段は、上記検査結果に基づいてオブジェクト置換の可否を判定する際に、削除対象のオブジェクトのサービスリストに登録されている全てのオブジェクトが、非実行状態であり、且つ、他のオブジェクトに送信したメッセージの返答を待っている状態でもない場合に、オブジェクト置換可と判定することを特徴とする請求項９記載のデータ処理装置。

【請求項１１】　上記オブジェクト置換手段は、オブジェクトの置換処理を行う際に、上記外部システムから取り込んだ新たなオブジェクトのサービスリストの内容を、上記一時記憶領域にコピーした内容に基づいて設定することを特徴とする請求項９記載のデータ処理装置。

【請求項１２】　上記実行主体を構成するオブジェクトは、当該オブジェクトに対してサービスを提供するオブジェクトが登録されたシステム依存リストを備え、

上記オブジェクト置換手段は、オブジェクトの置換処理を行う際に、削除対象のオブジェクトが登録されていたシステム依存リストの内容を更新し、それらのシステム

依存リストに対して上記外部システムから取り込んだ新たなオブジェクトを登録することを特徴とする請求項１記載のデータ処理装置。

【請求項１３】　上記外部システムは、伝送路を介して接続されるサーバシステムであり、上記オブジェクト置換手段によって取り込まれるオブジェクトが記憶された記憶媒体を備えることを特徴とする請求項１記載のデータ処理装置。

【請求項１４】　１つ以上のオブジェクトから構成される少なくとも１つの実行主体と、上記実行主体に対してサービスを提供する１つ以上のオブジェクトから構成される少なくとも１つの実行環境とを備えたデータ処理システムにおけるデータ処理方法において、

上記実行主体又は実行環境を構成するオブジェクトからの要求に基づいて、

上記実行環境を構成するオブジェクトを削除する削除ステップと、

外部システムから新たなオブジェクトを取り込む取込ステップとを行い、

上記実行環境を構成するオブジェクトを置換することを特徴とするデータ処理方法。

【請求項１５】　上記削除ステップは、

オブジェクトの削除要求を受けて、削除対象オブジェクトに対する処理の権利条件を検査する権利検査ステップと、

上記削除対象オブジェクトを削除するかどうかを決定する削除決定ステップと、

上記削除対象オブジェクトを削除するために、当該削除対象オブジェクトの占有する記憶領域を解放する解放ステップとを有することを特徴とする請求項１４記載のデータ処理方法。

【請求項１６】　上記権利検査ステップでは、上記削除対象オブジェクトの優先度を示すダウンロード受諾レベルと、上記削除要求を出したオブジェクトの優先度を示すダウンロード許可レベルとを比較し、所定の条件にあるかどうかを検査することを特徴とする請求項１５記載のデータ処理方法。

【請求項１７】　上記所定の条件は、上記削除要求を出したオブジェクトが備えるダウンロード許可レベルが、上記削除対象オブジェクトが備えるダウンロード受諾レベルより大きい場合であることを特徴とする請求項１６記載のデータ処理方法。

【請求項１８】　上記削除決定ステップでは、上記削除対象オブジェクトがサービスを提供するオブジェクトが登録されたサービスリストに存在する全てのオブジェクトの動作状態を検査し、所定の状態にある場合に、当該削除対象オブジェクトのサービスリストを一時記憶領域にコピーすることを特徴とする請求項１５記載のデータ処理方法。

【請求項１９】　上記所定の状態は、上記削除対象オブジェクトのサービスリスト中で参照されるオブジェクトの全てが、非実行状態であり、且つ、他のオブジェクトに送信したメッセージの返答を待っている状態でもない場合であることを特徴とする請求項１８記載のデータ処理方法。

【請求項２０】　上記解放ステップでは、上記データ処理システムの記憶領域から、上記削除対象オブジェクトを表現するデータを全て解放することにより、上記削除対象オブジェクトをデータ処理システムから削除することを特徴とする請求項１５記載のデータ処理方法。

【請求項２１】　上記取込ステップは、

オブジェクト取込要求を受けて、取込対象オブジェクトに対する処理の権利条件を検査する権利検査ステップと、

上記取込対象オブジェクトを上記外部システムより取り込み、上記データ処理システム内に記憶領域を確保して新たにオブジェクトを生成する生成ステップと、

新たに生成したオブジェクトを当該オブジェクトの実行環境に登録する登録ステップとを有することを特徴とする請求項１４記載のデータ処理方法。

【請求項２２】　上記権利検査ステップでは、上記取込対象オブジェクトの優先度を示すダウンロード受諾レベルと、オブジェクト取込要求を出したオブジェクトの優先度を示すダウンロード許可レベルとを比較し、所定の条件にあるかどうかを検査することを特徴とする請求項２１記載のデータ処理方法。

【請求項２３】　上記削除ステップの前に、オブジェクトの置換を要求したオブジェクトと、置換対象オブジェクトとを比較して、オブジェクトの置換を要求したオブジェクトが置換対象オブジェクトを置換する権限を有するか否かを検査し、権限を有する場合にだけ、上記削除ステップ及び取込ステップを行いオブジェクトを置換することを特徴とする請求項１４記載のデータ処理方法。

【請求項２４】　上記実行環境を構成するオブジェクトに、当該オブジェクトの優先度を示すダウンロード受諾レベルを備えさせるとともに、

上記実行主体を構成するオブジェクトに、当該オブジェクトの優先度を示すダウンロード許可レベルを備えさせ、

上記検査を行う際は、オブジェクトの置換を要求したオブジェクトのダウンロード許可レベルと、置換対象オブジェクトのダウンロード受諾レベルとを比較することで、オブジェクトの置換を要求したオブジェクトが置換対象オブジェクトを置換する権限を有するか否かを検査することを特徴とする請求項２３記載のデータ処理方法。

【請求項２５】　上記実行環境を構成するオブジェクトに、サービスを提供する対象のオブジェクトが登録されたサービスリストを備えさせ、

上記削除ステップの前に、削除対象のオブジェクトのサ

ービスリストに登録されたオブジェクトの状態を検査し、その検査結果に基づいてオブジェクト置換の可否を判定し、オブジェクト置換可と判定された場合にだけ、当該サービスリストの内容を一時記憶領域にコピーした上で、上記削除ステップ及び取込ステップを行いオブジェクトを置換することを特徴とする請求項１４記載のデータ処理方法。

【請求項２６】　上記判定を行う際は、削除対象のオブジェクトのサービスリストに登録されている全てのオブジェクトが、非実行状態であり、且つ、他のオブジェクトに送信したメッセージの返答を待っている状態でもない場合に、オブジェクト置換可と判定することを特徴とする請求項２５記載のデータ処理方法。

【請求項２７】　上記取込ステップでは、上記外部システムから取り込んだ新たなオブジェクトのサービスリストの内容を、上記一時記憶領域にコピーした内容に基づいて設定することを特徴とする請求項２５記載のデータ処理方法。

【請求項２８】　上記実行主体を構成するオブジェクトに、当該オブジェクトに対してサービスを提供するオブジェクトが登録されたシステム依存リストを備えさせ、上記取込ステップでは、上記削除ステップにおいて削除対象となったオブジェクトが登録されていたシステム依存リストの内容を更新し、それらのシステム依存リストに対して、上記外部システムから取り込んだ新たなオブジェクトを登録することを特徴とする請求項１４記載のデータ処理方法。

【請求項２９】　上記外部システムは、伝送路を介して接続されるサーバシステムであり、上記取込ステップで取り込まれるオブジェクトが記憶された記憶媒体を備えることを特徴とする請求項１４記載のデータ処理方法。

【請求項３０】　１つ以上のオブジェクトから構成される少なくとも１つの実行主体と、上記実行主体に対してサービスを提供する１つ以上のオブジェクトから構成される少なくとも１つの実行環境とを備えたデータ処理システムにおけるデータ処理プログラムを提供するプログラム提供媒体であって、

上記実行主体又は実行環境を構成するオブジェクトからの要求に基づいて、上記実行環境を構成するオブジェクトを削除するとともに外部システムから新たなオブジェクトを取り込んで、上記実行環境を構成するオブジェクトを置換する処理を実行するデータ処理プログラムを提供することを特徴とするプログラム提供媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、１つ以上のオブジェクトから構成される実行主体と、実行主体に対してサービスを提供するオブジェクトから構成される実行環境とを備えたデータ処理装置に関する。また、そのような実行主体及び実行環境を備えたデータ処理システムにおけるデータ処理方法に関する。また、そのようなデータ処理システムにおけるデータ処理プログラムを提供するプログラム提供媒体に関する。

【０００２】

【従来の技術】一般にデータ処理システムでは、実行環境（オペレーティングシステム等）の上で実行主体（アプリケーションプログラム等）が動作する。換言すれば、アプリケーションプログラム等の実行主体は、オペレーティングシステム等の実行環境が提供するサービスを用いて動作する。

【０００３】このようなデータ処理システムにおいて、通常、アプリケーションプログラムにサービスを提供するオペレーティングシステムは、どのようなアプリケーションプログラムを動作させるかを全て予測して設計されている訳ではない。そのため、計算資源を用いたサービスとして、全てのアプリケーションプログラムの性質に応じて最適化されたサービスを常に提供できるとは限らない。

【０００４】このことを、計算資源を用いたサービスとしてメッセージパッシング機構を提供するオペレーティングシステム上で動作する２種類のアプリケーションプログラムAppA, AppBを例に挙げて説明する。

【０００５】ここで、アプリケーションプログラムAppAは、データベースのサーバを構成するプログラミングモジュールと、２つのクライアントとからなるとする。また、これらのクライアントは、対等な優先度でサーバにアクセスするものとする。

【０００６】一方、アプリケーションプログラムAppBは、イーサネットからのデータを処理するクライアントと、シリアルラインからのデータを処理するクライアントと、クライアントからの要求に応じてデータ処理をするサーバとからなるとする。

【０００７】一般に、シリアルラインとイーサネットの動作には優先順位があり、シリアルラインで頻繁にメッセージのやり取りを行っているときでも、イーサネットから制御パケット（例えば、イーサネットから連続的にデータを読むため、一時的にシリアルラインからのデータの読み込みを停止するように要求する制御パケット）が来た場合には、当該制御パケットを優先的に処理することが要求される。制御パケットを優先的に処理するようにしないと、優先的に処理することが要求されている制御パケットの処理に待ち時間が必要となってしまう。

【０００８】以上のような２種類のアプリケーションプログラムのうち、アプリケーションプログラムAppAは、優先度の異なるメッセージがないので、メッセージの優先度チェックを行うような必要はない。したがって、メッセージの優先度チェックを行わない単純なメッセージパッシング機構を用いた方が、実行性能が向上する。

【０００９】一方、アプリケーションプログラムAppBでは、制御パケットを優先的に処理するための機構が必要

である。したがって、アプリケーションプログラムAppBに対して提供するメッセージパッシング機構は、メッセージの優先度をチェックする機構を備えている必要がある。

【００１０】しかしながら、メッセージの優先度チェックを行うようにすると、アプリケーションプログラムAppAを実行した場合に、アプリケーションプログラムAppAには不要な優先度チェックを行うこととなるので、アプリケーションプログラムAppAの実行性能が劣化する。一方、メッセージの優先度チェックを行わないメッセージパッシング機構を用いると、アプリケーションプログラムAppBを実行するときに、アプリケーションプログラムAppBの要求に応じた動作が不可能となってしまう。

【００１１】以上のように、２つ以上のプログラミングモジュールが通信して動作するアプリケーションプログラムに対して最適なサービスは、プログラミングモジュール間で行われる通信の性質に依存して異なっている。したがって、以上のような例では、プログラミングモジュール間で行われる通信の性質に応じて、メッセージパッシング機構のサービスを提供するオペレーティングシステムを変更可能であることが望まれる。

【００１２】ところで、以上のような問題は、アプリケーションプログラムを更新したり、新しいアプリケーションプログラムをダウンロードして使用するような場合に生じやすい。

【００１３】近年、パーソナルコンピュータ（PC:Personal Computer）やネットワークコンピュータ（NC:Network Computer）では、アプリケーションプログラムのダウンロード機能が標準装備されつつあり、そのような環境では、システム設計時に想定していない性質を持つアプリケーションプログラムがダウンロードされ実行される可能性が高い。このような環境で使用されるオペレーティングシステムでは、実行される全てのアプリケーションプログラムの性質を予測して設計しておくことは不可能である。

【００１４】また、家電製品（CE:Consumer Electronics）に使用される組み込み用のオペレーティングシステム（以下、組み込みＯＳと称する。）においても事情は同じである。従来の多くの家電製品は、ＣＤ－ＲＯＭやＭＤ等のリムーバブルメディアのサポートやネットワーク接続機能などが無く、アプリケーションプログラムを更新したり、新しいアプリケーションプログラムをダウンロードしたりする環境が未整備であった。そのため、従来の多くの家電製品では、組み込まれるアプリケーションプログラムに合わせて、オペレーティングシステムの機能を予め最適化しておくことが可能だった。

【００１５】しかしながら、近年、家電製品に関しても、リムーバブルメディアのサポートやネットワーク接続機能などを付加して、アプリケーションプログラムを更新したり、新しいアプリケーションプログラムをダウンロードしたりする環境が整備されつつある。それに伴い、家電製品に関しても、システム設計当時に想定していなかったアプリケーションプログラムの実行が可能になってきている。そのため、組み込みＯＳにおいても、全てのアプリケーションプログラムに対して、効率良く計算資源を用いたサービスを提供することが困難になりつつある。

【００１６】

【発明が解決しようとする課題】以上のように、従来のデータ処理システムでは、全てのアプリケーションプログラムの性質に応じて最適化したサービスを常に提供できるとは限らなかった。特に、システム設計後に更新されたりダウンロードされたりしたアプリケーションプログラムに対しては、それらのアプリケーションプログラムに合わせたサービスを提供するようなことが出来なかった。そこで、システム設計後に更新されたりダウンロードされたりしたアプリケーションプログラムに対しても、好適なサービスを提供できるようにすることが課題となっている。

【００１７】なお、想定しうるシステムサービスの機能を予め複数用意しておき、アプリケーションプログラムに選択させるという方法を用いれば、いくつかのアプリケーションプログラムに対しては好適なサービスを提供することが可能である。しかしながら、全てのアプリケーションプログラムに対して、満足のいく機能をサポートすることは不可能である。また、組み込みＯＳの場合は、一般にハードウェア上の搭載メモリ量に限りがあるため、プログラムサイズに制約がある。そのため、特に組み込みＯＳの場合、想定されるシステムサービスの機能を複数用意することは好ましくない。

【００１８】本発明は、以上のような従来の実情に鑑みて提案されたものであり、想定されるシステムサービスの機能を予め複数用意したりするようなことなく、実行主体に応じた好適な実行環境を提供することが可能なデータ処理装置及びデータ処理方法、並びにそのような機能を実現するデータ処理プログラムを提供するプログラム提供媒体を提供することを目的としている。

【００１９】

【課題を解決するための手段】本発明に係るデータ処理装置は、１つ以上のオブジェクトから構成される少なくとも１つの実行主体と、上記実行主体に対してサービスを提供する１つ以上のオブジェクトから構成される少なくとも１つの実行環境と、上記実行主体又は実行環境を構成するオブジェクトからの要求に基づいて、上記実行環境を構成するオブジェクトを削除するとともに外部システムから新たなオブジェクトを取り込んで、上記実行環境を構成するオブジェクトの置換処理を行うオブジェクト置換手段とを備えることを特徴とする。

【００２０】以上のような本発明に係るデータ処理装置では、実行環境を構成するオブジェクトをオブジェクト

置換手段により置換することで、様々な実行主体に対応するように実行環境を変更することができる。

【００２１】また、本発明に係るデータ処理方法は、１つ以上のオブジェクトから構成される少なくとも１つの実行主体と、上記実行主体に対してサービスを提供する１つ以上のオブジェクトから構成される少なくとも１つの実行環境とを備えたデータ処理システムにおけるデータ処理方法であり、上記実行主体又は実行環境を構成するオブジェクトからの要求に基づいて、上記実行環境を構成するオブジェクトを削除する削除ステップと、外部システムから新たなオブジェクトを取り込む取込ステップとを行い、上記実行環境を構成するオブジェクトを置換することを特徴とする。

【００２２】以上のような本発明に係るデータ処理方法では、削除ステップにより、実行環境を構成するオブジェクトが削除され、取込ステップにより、外部システムから新たなオブジェクトを取り込まれ、その結果、実行環境を構成するオブジェクトが置換される。このように、実行環境を構成するオブジェクトを置換することで、様々な実行主体に対応するように実行環境を変更することができる。

【００２３】また、本発明に係るプログラム提供媒体は、１つ以上のオブジェクトから構成される少なくとも１つの実行主体と、上記実行主体に対してサービスを提供する１つ以上のオブジェクトから構成される少なくとも１つの実行環境とを備えたデータ処理システムにおけるデータ処理プログラムを提供するプログラム提供媒体である。そして、上記実行主体又は実行環境を構成するオブジェクトからの要求に基づいて、上記実行環境を構成するオブジェクトを削除するとともに外部システムから新たなオブジェクトを取り込んで、上記実行環境を構成するオブジェクトを置換する処理を実行するデータ処理プログラムを提供することを特徴とする。

【００２４】なお、本発明において、オブジェクト取込元となる外部システムは、例えば、伝送路を介して接続されるサーバシステムであり、取り込まれるオブジェクトが記憶された記憶媒体を備える。

【００２５】

【発明の実施の形態】以下、本発明の実施の形態について、図面を参照しながら詳細に説明する。

【００２６】１．概略

以下に説明する本発明の実施の形態では、アプリケーションプログラムの性質に合わせて、オペレーティングシステムを構成するオブジェクト（以下、システムオブジェクトと称する。）をダウンロードしてオペレーティングシステムの構成を柔軟に変更し、アプリケーションプログラムの動作に好適な実行環境をカスタマイズできるようにする。

【００２７】すなわち、オブジェクト指向の概念を適用し、オブジェクトのダウンロードの機能をオペレーティングシステムにも適用する。そして、アプリケーションプログラムを構成するオブジェクト（以下、アプリケーションオブジェクトと称する。）のダウンロード時に、必要に応じてシステムオブジェクトもダウンロードしてオペレーティングシステムの構成を動的に柔軟に変更する。これにより、アプリケーションプログラムの性質に合わせて、当該アプリケーションプログラムの動作に好適な実行環境となるように、オペレーティングシステムをカスタマイズする。

【００２８】基本アーキテクチャとしては、オペレーティングシステムのサービス提供部分をオブジェクトとして提供するオブジェクト指向オペレーティングシステムを導入する。そして、アプリケーションプログラムの性質をよく知っているアプリケーションプログラマに対して、システムオブジェクトの仕様を公開する。アプリケーションプログラマは、アプリケーションプログラムを構成するアプリケーションオブジェクトの性質に応じてシステムオブジェクトをカスタマイズする。アプリケーションオブジェクトのダウンロード時に、カスタマイズ後のシステムオブジェクトを一緒にダウンロードし、アプリケーションプログラムを実行する。

【００２９】図１にシステムオブジェクトのダウンロードの例を示す。図１において、Ａ１，Ａ２，Ｂ１，Ｂ２はアプリケーションオブジェクトである。ＳｆｏｒＡは、Ａ１，Ａ２に対して最適化されたメッセージパッシングサービスを提供するシステムオブジェクトである。ＳｆｏｒＢは、Ｂ１，Ｂ２に対して最適化されたメッセージパッシングサービスを提供するシステムオブジェクトである。Ａ１，Ａ２を実行するとき、Ａ１，Ａ２をダウンロードするのと同時に、ＳｆｏｒＡをダウンロードする。Ｂ１，Ｂ２を実行するとき、Ａ１，Ａ２，ＳｆｏｒＡをアンロードした後に、Ｂ１，Ｂ２，ＳｆｏｒＢをダウンロードする。

【００３０】ところで、組み込みＯＳに対する要求事項のうち、最も重要な点の一つは、オペレーティングシステムの実装機構の単純化である。なぜなら、使用メモリ量を節約し、システムの構成価格を削減することが強く要求されるからである。オブジェクト指向の概念の適用は、次の意味でこの要求に貢献する。第一に、プログラムをダウンロードする場合に、様々な情報をオブジェクト内部に隠蔽しておくことで、複雑なオペレーションを単純化できる。第二に、システムオブジェクトとアプリケーションオブジェクトの基本的実行機構を共有できる。

【００３１】しかしながら、このような特徴を有するオブジェクト指向技術を適用するとしても、次の２つの問題に注意する必要がある。第１の問題は、システムオブジェクトとアプリケーションオブジェクトの相違に起因する問題であり、第２の問題は、組み込みＯＳ上での実現することに起因する問題とである。前者の問題で特に

注意しなければならないのは、システムオブジェクトは、アプリケーションオブジェクトと違い、オペレーティングシステムとしてのサービスを提供しなければならないので、アプリケーションオブジェクトの動作が停止しないように、変更は慎重に行う必要があるということである。また、後者の問題に対処するために、アプリケーションオブジェクトとシステムオブジェクトのダウンロード機構を共有化したり、システムオブジェクトの置換時に不必要なメモリ領域を削除して、使用メモリを削減する必要がある。

【００３２】２．システムオブジェクトダウンロードの具体的方法

システムオブジェクトダウンロードの方法のポイントは２つある。一つは、システムオブジェクトを安全にダウンロードする場合の必要事項が、アプリケーションオブジェクトをダウンロードする場合の必要事項とどのように異なり、それをどのように満足させるかを示すことである。もう一つは、組み込みＯＳにおいて有用な、システムオブジェクトのダウンロードの実現方法を示すことである。

【００３３】２．１　システムアーキテクチャの仮定

システムオブジェクトダウンロードを行うために、オペレーティングシステムが持つべきアーキテクチャの仮定について述べる。「オブジェクト」はアプリケーションプログラマが変更、追加、削除、置換が可能なプログラミングモジュールである。オブジェクト間ではメッセージパッシングによるデータ交換、起動が可能である。オブジェクトは、オペレーティングシステムのサービス提供モジュール、又はアプリケーションプログラムを構成するプログラムモジュールに成り得る。前者が「システムオブジェクト」、後者が「アプリケーションオブジェクト」である。

【００３４】オペレーティングシステムは、アプリケーションプログラマにより不可変の「カーネル」と、システムオブジェクトの集合で構成される。ここで、カーネルは、オブジェクト間のコンテキスト切り替えなどの必要最低限の機能を持つ。なお、「カーネル」は、システムによっては、マイクロカーネルやナノカーネルと呼ばれることもある。システムの安全な動作のため、システムオブジェクトは全てが置換可能であるとは限らない。

【００３５】システムオブジェクトとアプリケーションオブジェクトの最も大きな差異は、前者はアプリケーションオブジェクトの要求に応じて、サービスを提供することである。

【００３６】例えば、オブジェクト指向オペレーティングシステムでは、メッセージ送信要求やメモリ割当要求などが、オペレーティングシステムのサービスとして定義される。サービス要求は、システムコールなどの特殊命令により処理される。アプリケーションオブジェクトからのサービス要求は、カーネルによりディスパッチされ、システムオブジェクトが起動される。なお、ディスパッチとは、カーネルが、サービス要求を認識して、当該サービス要求をどのシステムオブジェクトに受け渡すかを判断し、サービス要求受け渡し先のシステムオブジェクトを起動し、当該システムオブジェクトにサービス要求を受け渡すことである。そして、サービス提供が終了すると、制御はアプリケーションオブジェクトに戻る。

【００３７】また、システムオブジェクトは、実行スタック、メッセージキュー、メモリセグメントといった実行に必要なプリミティブ構造体、すなわち計算資源を直接アクセスできるという点でも、アプリケーションオブジェクトとは異なっている。

【００３８】図２に仮定するシステムアーキテクチャの例を示す。Ａ１，Ａ２は互いにメッセージ通信するアプリケーションオブジェクトである。Ｓ１，Ｓ２，Ｓ３はシステムオブジェクトで、Ｓ３はメッセージパッシングサービスを提供する。Ａ１からＡ２にメッセージ送信するとき、メッセージ送信要求はカーネルにディスパッチされ、メッセージ送信サービスを提供するＳ３に制御が切り替わる。Ｓ３のサービス提供終了後、Ａ２が起動される。

【００３９】また、ダウンロードのサービスを提供するのもシステムオブジェクトである。ここでは、ダウンロードサービスを提供するシステムオブジェクトを「Downloader」と称する。アプリケーションオブジェクト又はシステムオブジェクトからのダウンロード要求が起こると、カーネルによりオブジェクト「Downloader」が起動され、ダウンロード処理が実行される。ダウンロード処理の具体的方策は後述する。

【００４０】各オブジェクトの使用するメモリ領域は、システムオブジェクトとアプリケーションオブジェクトの違いに関係なく、図３に示す通りである。オブジェクトの使用するメモリ領域には二種類ある。コンパイル時初期化メモリ領域は、コンパイラによって初期化値が決定される領域である。オブジェクトの実行メソッド、すなわち、実行命令の集合であるコード領域と、メソッド中で使用されるデータが格納されるデータ領域とがこれに含まれる。オブジェクトの生成時に、コンパイラに生成されたバイナリファイル中に初期化値が、それぞれの領域に格納される。コード領域に関してはＲＯＭ上に配置することも可能である。ＯＳ動作時初期化メモリ領域は、オペレーティングシステムのブート時に実行されるオペレーティングシステム初期化手続きによって初期値が決定される領域である。ＩＤはオブジェクトの識別子であり、実行スレッドは実行時のオブジェクトの状態を示す構造体である。実行スタックは実行時に使用されるスタックである。ヒープ用メモリ領域はメソッド中で(例えば、プログラミング言語Ｃ＋＋でのnewオペレーションにより)、動的に割り当てられるメモリ領域であ

り、ヒープ用に使用される。以上のメモリ領域は、ダウンロードに伴うオブジェクトの生成や削除の過程で、領域の割り当てや削除が行われる。

【００４１】２．２　システムオブジェクトダウンロードに対する要求事項

システムオブジェクトのダウンロードを行う場合に要求される事項は、大きく２つに分類される。一つは、アプリケーションオブジェクトをダウンロードするのではなく、システムオブジェクトがダウンロードの対象だからこそ特に要求される事項であり、もう一つは、特に組み込みＯＳということを意識した場合に要求される事項である。

【００４２】具体的には、システムオブジェクトはアプリケーションオブジェクトにサービスを提供するので、システムオブジェクトをダウンロードする際は、次の２つの要求を満たす必要がある。

【００４３】（１）ダウンロード要求証明

システムオブジェクトのダウンロードを全てのオブジェクトに許可することはできない。不適当なタイミングでのシステムオブジェクトの置換や悪意のある置換により、システムのサービス提供が不可能になり、最悪の場合には、システムが動作不能になる可能性があるからである。このような事態を避けるため、ダウンロードを要求するオブジェクトに対して「ダウンロード要求証明」を与えることにより、システムオブジェクトのダウンロードを要求可能なオブジェクトを制限する方法が必要である。

【００４４】（２）安全置換性

システムオブジェクトのダウンロードを開始した時、アプリケーションオブジェクトに対してサービスを提供中の可能性がある。この時、システムオブジェクトの置換により、そのアプリケーションオブジェクトの動作が保証されない可能性がある。例えば、メッセージ送信サービスを提供するシステムオブジェクトを置換している最中に、メッセージ送信要求が発生すると、誤動作する可能性がある。また、メッセージ送信サービスを提供するシステムオブジェクトを置換している最中に、メッセージ送信処理の途中のオブジェクトがある場合、置換されるシステムオブジェクトには、メッセージ送信サービスのための中間データ（例えば、メッセージキューに格納されているメッセージ）があるが、このデータを置換後の新しいシステムオブジェクトへ委譲する必要がある。このような問題を解決するために、「安全置換性」を確保できる仕組み必要である。

【００４５】また、組み込みＯＳ上でシステムオブジェクトのダウンロード機構を実現することを考慮すると、システムオブジェクトをダウンロードする際は、次の２つの要求を満たすことが望ましい。

【００４６】（１）ダウンロード機構の使用メモリ量を低減させるために、ダウンロード機構を単純化することが望ましい。そのため、システムオブジェクト及びアプリケーションオブジェクトでダウンロード機構を共有するように実装することが望ましい。

【００４７】（２）ダウンロードでオブジェジェクトの置換中に使用するメモリ量を低減することが望ましい。そのため、オブジェクトの置換中に必要な中間生成物の量を低減することが望ましい。

【００４８】２．３　要求事項に対する解決策

上述のようなシステムオブジェクトダウンロードに対する要求を満たしたダウンロード方法について説明する。

【００４９】２．３．１　ダウンロード受諾レベルとダウンロード許可レベル

２．２章で示した要求事項のうち、「ダウンロード要求証明」の要求を満たすために、「ダウンロード受諾レベル」と「ダウンロード許可レベル」を導入する。前者はダウンロードされるオブジェクトに設定され、後者はダウンロードの要求を出すオブジェクトに設定される。ダウンロード受諾レベルは、各オブジェクトに一つずつ設定される整数値で、そのオブジェクトのダウンロードの困難性を定義する。この値が大きければ、ダウンロードが難しくなる。システム設計者はシステムオブジェクトに適当なダウンロードレベルを設定する。基本的にアプリケーションオブジェクトのダウンロード受諾レベルは、システムオブジェクトのダウンロード受諾レベルより小さく設定される。一方、ダウンロードを要求するオブジェクトには、ダウンロード許可レベルが設定される。ダウンロード許可レベルはそのオブジェクトのダウンロードする能力を示す値である。このレベルもシステム設計者により適当な値が設定される。

【００５０】例えば、オブジェクトＡのダウンロード許可レベルがDa_Aで、オブジェクトＢのダウンロード受諾レベルがDp_Bの場合、オブジェクトＢがシステム中で動作中のオブジェクトＡを置換することを許可されるのは、下記の条件１を満たす場合である。

【００５１】Da_A ＞ Dp_B　　・・・条件１

例を図４に示す。アプリケーションオブジェクトＡ１のダウンロード許可レベルが１０であり、アプリケーションオブジェクトＢ１，Ｂ２のダウンロード受諾レベルが５であり、システムオブジェクトＣ１，Ｃ２のダウンロード受諾レベルが２０であるとする。このとき、Ａ１は、Ｂ２をダウンロードして、Ｂ１をＢ２に置換することが可能であるが、Ｃ２をダウンロードして、Ｃ１をＣ２に置換することはできない。

【００５２】２．３．２　システム依存リストとサービスリスト

２．２章で示した「安全置換性」の要求を満たすために、「システム依存リスト」と「サービスリスト」を導入する。あるアプリケーションオブジェクトとシステムオブジェクトの間には、サービス依頼者／サービス提供者としての相対関係がある。この相対関係を知ること

は、ダウンロードされ置換されるシステムオブジェクトが、どのアプリケーションオブジェクトと関係があるかを調べるのに有用である。この相対関係を表現するのが、システム依存リストとサービスリストである。サービス依頼者からサービス提供者を参照する場合、サービス依頼者に保持されるシステム依存リストが用いられる。一方、サービス提供者からサービス依頼者を参照するには、サービス提供者に保持されるサービスリストが用いられる。

【００５３】あるサービス依頼者であるアプリケーションオブジェクトの動作は、システムオブジェクトで提供されるサービスの集合としても定義される。ここで、システムオブジェクトがどのアプリケーションオブジェクトにサービスを提供しているかを示すリンクがシステム依存リストである。システム依存リストＤｌは、次のように定義される。

【００５４】システム依存リストの定義

Dl＝{<Index_1, Lst_1>, <Index_2, Lst_2>,..., <Index_n, Lst_n>}

Index_n:サービスのインデックス

Lst_n　:サービスを提供するシステムオブジェクトのIDの集合

Index_nは、オペレーティングシステムが提供するサービスのインデックスである。Lst_nは、サービスを提供するシステムオブジェクトのリストである。システム依存リストは、全てのオブジェクトに対して定義される。このリンクは、アプリケーションオブジェクトにおいてサービス要求があった場合、カーネルがシステムオブジェクトを起動する際にも用いられる。

【００５５】なお、システム依存リストは、基本的にはアプリケーションプログラマがアプリケーションオブジェクトを定義するときに、どのサービスを提供されたいかを考慮して定義する。しかしながら、アプリケーションプログラマが全てのシステムサービスを把握するのは負担になるので、システムのサービスを定義するシステム管理者によって予め定義されたシステム依存リストを容易に選択使用する仕組みを提供することが望ましい。

【００５６】システムオブジェクトがダウンロードにより置換される場合、システム依存リストは、利用されると同時に更新される必要がある。なぜなら、そこに含まれるシステムオブジェクトのＩＤが変更される可能性があるからである。更新のタイミングは、２．３．３で述べる。

【００５７】一方、サービス提供者であるシステムオブジェクトは、次のように定義されるサービスリストＳＩを持つ。

【００５８】サービスリストの定義

Sl＝{<ID_1, Dl_1>, <ID_2, Dl_2>, ..., <ID_n, Dl_n>}

ID_n:アプリケーションオブジェクトの識別子

Dl_n:アプリケーションオブジェクトのシステム依存リスト

サービスリストは、システムオブジェクトをダウンロードして既存のシステムオブジェクトを置換する場合に、そのシステムオブジェクトのサービスが続行中であるか、また、システムオブジェクトへのサービス提供要求中かをチェックするのに用いられる。すなわち、オブジェクト「Downloader」は、置換されるシステムオブジェクトのサービスリストを調査し、そのシステムオブジェクトがサービスを提供するアプリケーションオブジェクトの存在を調べる。アプリケーションオブジェクトが存在した場合は、例えば、置換を失敗させるとか、置換を延期するなどの処置を行う。サービスリスト内のシステム依存リストは、各アプリケーションオブジェクトの持つシステム依存リストが正しいものであることをチェックするためにも用いられる。

【００５９】あるアプリケーションオブジェクトがシステム上にインストールされるとき、そのアプリケーションオブジェクトがどのシステムオブジェクトにサービスを提供されるかは、システム依存リストの選択により決定される。オブジェクト「Downloader」は、システム依存リストを解析し、関連するシステムオブジェクトのサービスリストに、インストールされるアプリケーションオブジェクトのIDとシステム依存リストが登録される。システムオブジェクトがどのオブジェクトにもサービスを提供していないならば、サービスリストはnullである。

【００６０】図５にシステム依存リストとサービスリストの例を示す。図５の例では、メッセージ送信のサービスのインデックスが0x10で、「Mailer」という名前のシステムオブジェクトがそのサービスを提供する。また、メモリ割り当てサービスのインデックスが0x20で、「MemMgr」という名前のシステムオブジェクトがメモリ割り当てサービスを提供する。アプリケーションオブジェクト「ObjectA」のシステム依存リストDl_aは、{<0x10, {Mailer}>, <0x20, {MemMgr}>,...}のようになる。カーネルは、メッセージ送信要求をディスパッチしたら、システム依存リストDl_aをたどってオブジェクト「Mailer」を起動する。

【００６１】この例において、オブジェクト「Mailer」を置換する際、オブジェクト「Downloader」は、オブジェクト「Mailer」のサービスリストSl_Mailer＝{<ID_a, Dl_a>, <ID_b, Dl_b>}（ただし、ID_aはObjectAの識別子、ID_bはObjectBの識別子）をチェックする。そして、アプリケーションオブジェクトの存在の有無を調べ、サービス提供中かをチェックする。提供中でない場合は、サービス要求が来ないように処理をする。アプリケーションオブジェクト「ObjectA」とシステムオブジェクト「Mailer」を連続して置換する場合、「Mailer」が置換される前に、「ObjectA」が既に削除されていれば、「M

ailer」は問題なく削除、置換が可能である。

【００６２】２．４　置換アルゴリズム

つぎに、システムオブジェクトをダウンロードして、既存のシステムオブジェクトを置換するアルゴリズムについて説明する。以下に説明するアルゴリズムは、「ダウンロード要求証明」及び「安全置換性」の要求事項を満たすものである。

【００６３】２．３章で述べたシステムオブジェクトのダウンロード方法は、アプリケーションオブジェクトのダウンロードと同時に使用できる。アプリケーションオブジェクトのダウンロードに連動させて、アプリケーションの動作の性質に合わせたシステムオブジェクトをダウンロードすれば、アプリケーションオブジェクトの動作をカスタマイズすることが可能になる。アプリケーションオブジェクトのダウンロードとシステムオブジェクトのダウンロードは、同一アルゴリズムによって与えられる。これにより、組み込みＯＳとして考えた場合、可能な限りのダウンロード実現部分をアプリケーションオブジェクトとシステムオブジェクトで共有することが可能になる。

【００６４】しかしながら、オブジェクト「Downloader」の内部動作において、アプリケーションオブジェクトとシステムオブジェクトの区別をする方法が必要になる。このための一つの方法としては、それぞれのシステム依存リストがアプリケーションオブジェクト用のものか、システムオブジェクト用のものかを予め決めておくことが考えられる。この方法だと、余計な判別用フラグをオブジェクト内部情報として定義する必要がなくなり、メモリ使用量節約に有用である。

【００６５】システムオブジェクトの置換過程は次の２つのフェーズからなる。

【００６６】（１）アンロードフェーズ

アンロードフェーズでは、古いオブジェクトを削除する。アンロードフェーズは、削除フェーズと解放フェーズの２つのフェーズからなる。削除フェーズでは、古いシステムオブジェクトの動作を無効にし、解放フェーズでは、古いシステムオブジェクトが使用中のデータ領域を解放する。

【００６７】（２）ロードフェーズ

ロードフェーズでは、新しいオブジェクトを生成する。ロードフェーズは、割当フェーズと生成フェーズの２つのフェーズからなる。割当フェーズでは、新しいシステムオブジェクトが使用するデータ領域を割り当てて初期化し、生成フェーズでは、新しいシステムオブジェクトの動作を有効にする。

【００６８】以下、これらの各フェーズの詳細について説明する。

【００６９】２．４．１　アンロードフェーズ

アンロードフェーズでは、新しいシステムオブジェクトをダウンロードする準備として、既存のシステムオブジェクトの動作を無効にし、削除されるオブジェクトの使用メモリ領域を解放する。

【００７０】まず、アンロードフェーズの前半部分である削除フェーズについて、図６を参照して説明する。

【００７１】削除フェーズでは、先ず、ステップＳ１において、上述した条件１をチェックする。すなわち、アンロード要求をしたオブジェクトのダウンロード許可レベルが、アンロードされるシステムオブジェクトのダウンロード受諾レベルよりも大きいかを調べる。そして、条件１を満たさない場合は、エラーを返し、アンロードフェーズを中止する。

【００７２】次に、ステップＳ２において、削除されるオブジェクトがシステムオブジェクトかアプリケーションオブジェクトかを判別する。オブジェクトがアプリケーションオブジェクトの場合は削除フェーズを終了し、システムオブジェクトの場合はステップＳ３へ進む。

【００７３】ステップＳ３では、動作中オブジェクトのチェックを行う。オブジェクト「Downloader」は、削除するシステムオブジェクトのサービスリストがnullかどうかをチェックする。nullの場合は削除フェーズを終了し、サービスリストがnullでない場合はステップＳ４へ進む。

【００７４】ステップＳ４では、サービスリストに登録されているアプリケーションオブジェクトに対して、サービスリストにあるシステム依存リストを正しく保持しているかをチェックする。システム依存リストを正しく保持していない場合は、エラーを返し、アンロードフェーズを中止する。

【００７５】次に、ステップＳ５において、サービスリストに登録されているアプリケーションオブジェクトの動作状態をチェックする。なお、アプリケーションオブジェクトの動作状態には、「待機状態（Dormant）」、「動作中（Running）」、「返答待ち（Waiting）」、「無効状態（Free）」がある得る。「待機状態」は、他のオブジェクトからメッセージの受付が可能な状態であり、「動作中」は、他のオブジェクトから受け取ったメッセージに基づいて処理を行っている状態であり、「返答待ち」は、他のオブジェクトに送信したメッセージの返答を待っている状態であり、「無効状態」は、非実行状態である。なお、「無効状態」では、他のオブジェクトからのメッセージ受付も不可である。

【００７６】そして、ステップＳ５でのチェックの結果、動作状態が「動作中」又は「返答待ち」のアプリケーションオブジェクトがある場合は、ステップＳ６へ進む。

【００７７】ステップＳ６では、システムオブジェクトの削除を延期するように、システムオブジェクトの一つであるスケジューラに依頼する。スケジューラは、スケジューリングを行うシステムオブジェクトであり、オブジェクトの動作終了時に起動するメソッドを登録でき

る。そこで、システムオブジェクト削除延期の依頼を受けたスケジューラは、オブジェクトリストに動作中のアプリケーションオブジェクトを登録し、当該アプリケーションオブジェクトの動作終了時にアンロードフェーズが再開するようにする。

【００７８】一方、ステップＳ５でのチェックの結果、動作状態が「動作中」又は「返答待ち」のアプリケーションオブジェクトがない場合は、ステップＳ７へ進む。

【００７９】ステップＳ７では、削除を要求されたオブジェクトの動作状態を「無効状態」にする。この時点で、このオブジェクトはスケジューリングされることがなくなり、メッセージ受信もしなくなる。

【００８０】次に、ステップＳ８において、システムオブジェクトのサービスリストを新しいオブジェクトに受け渡すため、削除対象のシステムオブジェクトのサービスリストをオブジェクト「Downloader」に登録する。

【００８１】次に、ステップＳ９において、削除対象のオブジェクトの内部状態を、新しいオブジェクトに譲渡できるようにするために、削除対象のオブジェクトの内部状態を、オブジェクト「Downloader」に登録する。また、新しいオブジェクトに譲渡するデータもオブジェクト「Downloader」に登録する。ここで、新しいオブジェクトに何を譲渡するかについての情報や、新しいオブジェクトにそれらを受け渡すために起動されるメソッドは、予めオブジェクト「Downloader」に登録しておく。なお、このステップＳ９の処理は必須ではなく、システムオブジェクトの種類によって、内部状態等の譲渡が必要な場合にだけ行う。

【００８２】以上で削除フェーズが完了し、次に解放フェーズに移る。アンロードフェーズの後半部分である解放フェーズについて、図７を参照して説明する。

【００８３】解放フェーズでは、先ず、ステップＳ２１において、削除対象のオブジェクトのＩＤを削除するとともに、実行スレッド、実行スタック、ヒープ用のメモリ領域を解放する。

【００８４】次に、ステップＳ２２において、削除対象のオブジェクトのコード領域及びデータ領域を解放する。

【００８５】最後に、ステップＳ２３において、オブジェクトが削除されたことを、予め登録されている他のオブジェクトに通知する。これは、削除されたオブジェクトへの参照を削除したり、内部データの再初期化を行うためである。

【００８６】以上で、古いシステムオブジェクトの動作を無効にする削除フェーズと、古いシステムオブジェクトが使用中のデータ領域を解放する解放フェーズとが完了し、古いオブジェクトを削除するアンロードフェーズが完了する。

【００８７】なお、組み込みシステムの性質を考慮した場合、アンロードフェーズにおいて、動作中のオブジェクトの有無のチェック（削除フェーズのステップＳ５）は、必ずしも必要ではない。例えば、置換するシステムオブジェクトからサービス提供を受けるアプリケーションオブジェクトをアプリケーションプログラマの責任で全て停止するようにしてもよい。もしくは、削除した後にシステムオブジェクトの動作を無効にするようにしてもよい。この方法では、置換中に新旧の２つのシステムオブジェクトが同時存在することがなくなるので、置換に必要なメモリ量が節約できる。さらに、動作中のアプリケーションオブジェクトの処理をするコードが不必要になることも、メモリ量の節約になる。

【００８８】なお、動作中のアプリケーションオブジェクトを処理する手続きが必要か否かを決定する条件は、アプリケーションオブジェクトを停止させるために必要なプログラミングの負担と、必要なメモリ量とのトレードオフになる。

【００８９】また、システムオブジェクトの内部状態の情報等の受け渡し（削除フェーズのステップＳ９）を禁止し、オブジェクトの内部状態の情報等の受け渡しの処理を行うコードを削除しても、メモリ量を節約可能である。

【００９０】このように、削除フェーズのアルゴリズムは、図６に示した例に限定されるものではない。したがって、システムのコンフィギュレーションや性質により、削除フェーズのアルゴリズムを選択できるようにしておくことが望ましい。

【００９１】２．４．２　ロードフェーズ

ロードフェーズでは、新しいオブジェクトに必要なメモリ領域を確保し、オブジェクトをダウンロードする。

【００９２】まず、ロードフェーズの前半部分である割当フェーズについて、図８を参照して説明する。

【００９３】割当フェーズでは、先ず、ステップＳ１において、上述した条件１をチェックする。すなわち、アンロード要求をしたオブジェクトのダウンロード許可レベルが、アンロードされるシステムオブジェクトのダウンロード受諾レベルよりも大きいかを調べる。そして、条件１を満たさない場合は、エラーを返し、ロードフェーズを中止する。

【００９４】なお、本例では、ここでエラーが返されたとしても、システムではアンロードフェーズのキャンセルをしないようなアルゴリズムとしている。したがって、アプリケーションプログラマは、生成されるオブジェクトが条件１を満たすことの確認を、アンロードフェーズの前に確実に行う必要がある。システムがアンロードフェーズのキャンセル処理を行わない理由は、生成されるオブジェクト用のメモリ領域をロードフェーズ前に確保するために、削除されるオブジェクトの使用するメモリ領域を解放するという仕様にしているためである。

【００９５】条件１を満たしている場合は、ステップＳ３２に進み、新しく生成するオブジェクトのデータ領

域及びコード領域の割り当てを行う。

【００９６】次に、ステップＳ３３において、ダウンロード元のソースメディア（ネットワーク上のファイルシステム、或いはリムーバブルメディアやＦｌａｓｈＲＯＭなどの記録媒体）から、新たに生成するオブジェクトの内容をコピーする。

【００９７】次に、ステップＳ３４において、新しく生成するオブジェクトに対して、ＩＤ、実行スレッド、実行スタック、ヒープ用メモリ領域を割り当てる。

【００９８】以上で割当フェーズが完了し、次に生成フェーズに移る。ロードフェーズの後半部分である生成フェーズについて、図９を参照して説明する。

【００９９】生成フェーズでは、先ず、ステップＳ４１において、生成するオブジェクトがシステムオブジェクトかアプリケーションオブジェクトかを判別する。生成するオブジェクトがシステムオブジェクトの場合は、ステップＳ４２へ進む。なお、生成するオブジェクトがアプリケーションオブジェクトの場合は生成フェーズを終了する。

【０１００】ステップＳ４２では、アンロードフェーズで削除されたシステムオブジェクトのサービスリストを、生成するオブジェクトのサービスリストとして登録する。換言すれば、古いシステムオブジェクトのサービスリストを、新しいシステムオブジェクトに譲渡する。

【０１０１】次に、ステップＳ４３において、ステップＳ４２で譲渡されたサービスリストに基づいて、生成されたシステムオブジェクトがサービスを提供するアプリケーションオブジェクトが存在するかをチェックする。そして、存在する場合は、ステップＳ４４へ進み、存在しない場合は、ステップＳ４５へ進む。

【０１０２】ステップＳ４４では、生成されたシステムオブジェクトがサービスを提供する各アプリケーションオブジェクトについて、システム依存リスト中のシステムオブジェクトのＩＤを更新する。すなわち、それらのアプリケーションオブジェクトのシステム依存リスト中の古いシステムオブジェクトのＩＤを、新しく生成するシステムオブジェクトのＩＤに置き換える。その後、ステップＳ４５へ進む。

【０１０３】ステップＳ４５では、オブジェクト「Downloader」に削除されたシステムオブジェクトの内部状態が登録されているかをチェックする。登録されていない場合は、生成フェーズを終了し、登録されている場合は、ステップＳ４６へ進む。

【０１０４】ステップＳ４６では、削除されたシステムオブジェクトの内部状態を生成されたシステムオブジェクトに譲渡する。換言すれば、古いシステムオブジェクトから譲渡される内部データを新しいシステムオブジェクトに受け渡す。

【０１０５】以上で、新しいシステムオブジェクトが使用するデータ領域を割り当て初期化する割り当てフェーズと、新しいシステムオブジェクトの動作を有効にする生成フェーズとが完了し、新しいオブジェクトを生成するロードフェーズが完了する。

【０１０６】２．５　ダウンロード用アプリケーションインタフェース

アプリケーションオブジェクトのダウンロードとシステムオブジェクトのダウンロードは、同一のアプリケーションインタフェース（ＡＰＩ：Application Program Interface）によって記述される。具体的には、本例では、システムオブジェクト及びアプリケーションオブジェクトのダウンロード用のＡＰＩとして、下記に示す２つのＡＰＩ「Load()」「Unload()」を提供する。なお、これらのＡＰＩは、アプリケーションオブジェクトの内部、及びシステムオブジェクトの内部で使用可能である。

【０１０７】error Load (Symbol objectName);
error Unload (Symbol objectName);
Load()は、オブジェクトをダウンロードするためのＡＰＩである。置換の手続きの中では、ロードフェーズがLoad()の呼び出しにより実行される。Load()が呼び出されると、引数「objectName」で指定されるオブジェクトがダウンロードされる。アプリケーションオブジェクトとシステムオブジェクトは基本的に同一名前空間に存在する。ただし、アプリケーションオブジェクトをダウンロードする場合は、システム依存リストの更新等の処理（生成フェーズのステップＳ４２～Ｓ４６）は省略される。

【０１０８】Unload()は、オブジェクトをアンロードするためのＡＰＩ、すなわち既に存在しているオブジェクトを削除するためのＡＰＩである。置換の手続きの中では、アンロードフェーズがUnload()の呼び出しにより実行される。Unload()が呼び出されると、引数「objectName」で指定されるオブジェクトがアンロードされる。アンロードの対象になるものは、アプリケーションオブジェクトでもシステムオブジェクトでもよい。ただし、アプリケーションオブジェクトのアンロードの場合は、システム依存リストのチェック等の処理（削除フェーズのステップＳ３～Ｓ９）は省略される。

【０１０９】上記２つのＡＰＩは共にアプリケーションオブジェクト内で用いることができるので、アプリケーションオブジェクトに都合の良いタイミングで、アプリケーションオブジェクトやシステムオブジェクトをダウンロードすることが可能である。置換を行う場合の一連の動作は、Unload()とLoad()の連続使用により実現できる。

【０１１０】例えば、２つのアプリケーションオブジェクトAppA, AppBがあるとする。このとき、一方のアプリケーションオブジェクトAppAの性質に応じて最適化されたシステムオブジェクトSforAと、他方のアプリケーションオブジェクトAppBの性質に応じて最適化されたシス

テムオブジェクトSforBとを用意しておく。そして、AppA, AppBをダウンロードするアプリケーションオブジェクトをAppCとする。以下のプログラムは、この例におけるオブジェクトAppCのプログラム例である。

【０１１１】

```
AppC::LoadAppA ()
{   Load ("SforA");
    Load ("AppA");
}
AppC::ReplaceAppA ()
{   Unload ("AppA");
    Unload ("SforA");
    Load ("SforB");
    Load ("AppB");
}
```

AppCは、AppAをダウンロードする前に、LoadAppA()メソッド中で、Load() を用いてSforAをロードする。これにより、AppAには最適なシステムサービスが提供される。その後、AppBを実行するときには、ReplaceAppA()メソッドで、AppA及びSforAをアンロードし、AppB及びSforBをダウンロードする。これにより、AppBにも最適なシステムサービスが提供される。AppBをダウンロードする時に、古いアプリケーションオブジェクトAppAと、AppBの実行には不必要なシステムオブジェクトSforAとは、Unload() を用いてアンロードしておく。これにより、AppBの実行時や置換時に不必要なメモリを消費することがなくなる。

【０１１２】３．アプリケーションオブジェクトのカスタマイズの例と効果

システムオブジェクトダウンロードを用いたアプリケーションオブジェクト実行のカスタマイズの例と、その効果について説明する。

【０１１３】３．１　例１：メッセージパッシング方法の改善

アプリケーションオブジェクトAppAは、３つのアプリケーションオブジェクトAO-a1, AO-a2, AO-a3からなり、それらのアプリケーションオブジェクト間では対等の優先度及び頻度のメッセージパッシングを行うとする。このアプリケーションオブジェクトAppAの実行性能の観点からは、FIFOキューによる単純なメッセージパッシング機構が必要とされる。

【０１１４】また、アプリケーションオブジェクトAppBも、３つのアプリケーションオブジェクトAO-b1, AO-b2, AO-b3からなるとする。そして、多くの場合、AO-b1とAO-b2との間のメッセージパッシングは、AO-b1とAO-b3との間のメッセージパッシングよりも優先的にスケジューリングされるとする（なお、これは、メッセージパッシング機構ではなく、スケジューリング機構的に行われる。）。また、AO-b1とAO-b3との間には、優先度の高いメッセージパッシングが時々行われるとする。したがって、このアプリケーションオブジェクトAppBに対しては、優先度に応じてメッセージを処理するメッセージパッシング機構が必要とされる。アプリケーションオブジェクトAppAで用いられる単純なメッセージパッシング機構を、アプリケーションオブジェクトAppBが用いると、高優先度メッセージの処理に遅延が起こり、実行性能が劣化する。

【０１１５】また、単純なメッセージパッシング機構を持つシステムオブジェクトをMO1とし、優先度に応じてメッセージを処理するメッセージパッシング機構を持つシステムオブジェクトをMO2とする。そして、アプリケーションオブジェクトAppAとアプリケーションオブジェクトAppBは同時に動作せず、これらのアプリケーションオブジェクトAppA, AppBは必要に応じて他のアプリケーションオブジェクトAppCによってダウンロードされ実行されるとする。

【０１１６】アプリケーションオブジェクトAppCは、AppAをダウンロードするとき、AO-a1, AO-a2, AO-a3をダウンロードする前に、MO1をダウンロードする。このようにAO-a1, AO-a2, AO-a3及びMO1をダウンロードしたシステムの状態を示したのが図１０である。このとき、AppBが存在している場合は場合は、AppBを構成しているアプリケーションオブジェクトAO-b1, AO-b2, AO-b3をアンロードし、続いてMO2をアンロードする。

【０１１７】また、アプリケーションオブジェクトAppCは、AppBをダウンロードするとき、AO-b1, AO-b2, AO-b3をダウンロードする前に、MO2をダウンロードする。このようにAO-b1, AO-b2, AO-b3及びMO2をダウンロードしたシステムの状態を示したのが図１１である。このとき、AppAが存在している場合は、AppAを構成しているアプリケーションオブジェクトAO-a1, AO-a2, AO-a3をアンロードし、続いてMO1をアンロードする。

【０１１８】以上のようにダウンロード機能を用いることで、性質の異なる２つのアプリケーションオブジェクトAppA, AppBのそれぞれに対して、最適なメッセージパッシング機構を用いることが可能になる。

【０１１９】以下のプログラムは、アプリケーションオブジェクトAppCが、アプリケーションオブジェクトAppAをダウンロードするときに用いるメソッドLoadAppA()のプログラム例である。

【０１２０】

```
AppC::LoadAppA {
        Unload ("AO-b1");
        Unload ("AO-b2");
        Unload ("AO-b3");
        Unload ("MO2");
        Load ("MO1");
        Load ("AO-a1");
        Load ("AO-a2");
        Load ("AO-a3");
}
```

ここで、アプリケーションオブジェクトAppCのダウンロード許可レベルDa_appcと、システムオブジェクトMO1のダウンロード受諾レベルDp_mo1と、システムオブジェクトMO2のダウンロード受諾レベルDp_mo2との間には、上述した条件１を満たすように、Da_appc＞Dp_mo1、且つ、Da_appc＞Dp_mo2の関係が成り立つようにしておく。

【０１２１】なお、上記の例において、システムオブジェクトMO1のサービスリストをSl_mo1とし、システムオブジェクトMO2のサービスリストをSl_mo2とし、アプリケーションオブジェクトAO-a1,AO-a2,AO-a3のシステム依存リストをそれぞれDl_AO-a1,Dl_AO-a2,Dl_AO-a3とし、アプリケーションオブジェクトAO-b1,AO-b2,AO-b3のシステム依存リストをそれぞれDl_AO-b1,Dl_AO-b2,Dl_AO-b3としたとき、それらはそれぞれ以下のように記述される。

【０１２２】Sl_mo1＝{<ID_AO-a1,Dl_AO-a1>,<ID_AO-a2,Dl_AO-a2>,<ID_AO-a3,Dl_AO-a3>}

Dl_AO-a1 = {<0x10, {MO1}>...}

Dl_AO-a2 = {<0x10, {MO1}>...}

Dl_AO-a3 = {<0x10, {MO1}>...}

Sl_mo2＝{<ID_AO-b1,Dl_AO-b1>,<ID_AO-b2,Dl_AO-b2>,<ID_AO-b3,Dl_AO-b3>}

Dl_AO-b1 = {<0x10, {MO2}>...}

Dl_AO-b2 = {<0x10, {MO2}>...}

Dl_AO-b3 = {<0x10, {MO2}>...}

【０１２３】３．２　例２：デバッグモード

ダウンロード機構をデバッグモードに適用した例を図１２を用いて説明する。

【０１２４】ここでは、ダウンロードして実行されるアプリケーションオブジェクトAOをデバッグすることを考える。そして、デバッグのために、メッセージパッシングのモニタリングを試みるとする。

【０１２５】メッセージパッシングのモニタリングを試みるには、メッセージパッシング機構を変更してメッセージダンプ機能を入れ、プリンタやディスプレイ等の出力装置にモニタリングの結果を出力するようにする必要がある。

【０１２６】そこで、図１２に示すように、アプリケーションオブジェクトAOをダウンロードするタイミングで、デバッグ機能を持つメッセージパッシング機構を持つシステムオブジェクトMO2をダウンロードして、通常のメッセージパッシング機構を持つシステムオブジェクトMO1をMO2に置換する。そして、アプリケーションオブジェクトAOのデバッグの終了後、MO2をMO1に再置換する。

【０１２７】このようにすれば、全てのアプリケーションオブジェクトのメッセージパッシングがモニタリングされるのではなく、デバッグ対象のオブジェクトのメッセージパッシングのみがモニタリングされるため、デバッグが容易になる。

【０１２８】３．３　例３：適応的ネットワークプロトコル

２つの異なるネットワークプロトコルを使用して、リモートプログラムと通信する二種類のアプリケーションプログラムApp1,App2を考える。ここで、App1はネットワークプロトコルとしてUDP/IP (User Datagram Protocol／Internet Protocol) を用い、App2はネットワークプロトコルとしてDSM-CC (Digital StorageMedia Command and Control) を用いるとする。ネットワークプロトコルの実装は一般的にサイズが大きいので、組み込みＯＳに予め多くのプロトコルを実装しておくのは困難である。したがって、このような場合、本発明を適用したダウンロード機構は非常に有効である。

【０１２９】ダウンロード機構を適用した場合の例を図１３を用いて説明する。図１３に示すように、App1はアプリケーションオブジェクトAO1で構成され、App2はアプリケーションオブジェクトAO2で構成されるとする。また、UDP/IPを実装するシステムオブジェクトをNO1とし、DSM-CCを実装するシステムオブジェクトをNO2とする。そして、AO1をダウンロードする場合は、AO1をダウンロードする前にAO2をアンロードし、その後、AO1をダウンロードする。また、AO2をダウンロードする場合は、AO2をダウンロードする前にAO1をアンロードし、その後、AO2をダウンロードする。

【０１３０】このようにネットワークプロトコルのサービスを提供するシステムオブジェクトのダウンロードを行うことにより、アプリケーションオブジェクトが必要なネットワークプロトコルを常に用意することが可能である。

【０１３１】３．４　例４：適応的実行環境

単一のサービスだけでなく、アプリケーションオブジェクトの動作する動作環境全体を変更したい場合がある。そのような場合の例を、図１４を用いて説明する。

【０１３２】アプリケーションプログラムAppAは、並行オブジェクト指向モデルで定義される実行環境で動作し、アプリケーションオブジェクトAO-a1, AO-a2から構成されるとする。そして、AppA用の実行環境を定義するシステムオブジェクトとして「Mailer1」「MemMgr1」「ObjectCreator」があるとする。なお、「Mailer1」は

メッセージパッシング機構を提供するシステムオブジェクトであり、「MemMgr1」はメモリ管理機構を提供するシステムオブジェクトであり、「ObjectCreator」はオブジェクト生成/削除機構を提供するシステムオブジェクトである。

【０１３３】また、アプリケーションプログラムAppBは、マルチスレッド使用の動作環境で動作し、アプリケーションオブジェクトAO-b1,AO-b2から構成されるとする。そして、AppB用の実行環境を定義するシステムオブジェクトとして、「Mailer2」「MemMgr2」「ThreadCreator」があるとする。なお、「Mailer2」はスレッド間通信機構を提供するシステムオブジェクトであり、「MemMgr2」はメモリ管理機構を提供するシステムオブジェクトであり、「ThreadCreator」はスレッド生成/削除機構を提供するシステムオブジェクトである。

【０１３４】本例では、アプリケーションプログラムAppAが動作するとき、「Mailer2」「MemMgr2」「ThreadCreator」は不必要であり、また、アプリケーションプログラムAppBが動作するとき、「Mailer1」「MemMgr1」「ObjectCreator」は不必要である。

【０１３５】そこで、図１４に示すように、アプリケーションプログラムAppAをダウンロードして実行するときは、「AppB」「Mailer2」「MemMgr2」「ThreadCreator」をアンロードした上で、「AppA」「Mailer1」「MemMgr1」「ObjectCreator」をダウンロードし、「Mailer1」「MemMgr1」「ObjectCreator」を含むシステム上で、アプリケーションプログラムAppAを実行する。

【０１３６】同様に、アプリケーションプログラムAppBをダウンロードして実行するときは、「AppA」「Mailer1」「MemMgr1」「ObjectCreator」をアンロードした上で、「AppB」「Mailer2」「MemMgr2」「ThreadCreator」をダウンロードし、「Mailer2」「MemMgr2」「ThreadCreator」を含むシステム上で、アプリケーションプログラムAppBを実行する。

【０１３７】なお、オブジェクトのダウンロードを管理するアプリケーションオブジェクトをAppCとすると、例えばAppAをロードする際、アプリケーションオブジェクトAppCは、アプリケーションプログラムインターフェース「Load()」「Unload()」を用いた下記のようなメソッドAppC::LoadAppA()を実行する。

【０１３８】


```
AppC::LoadAppA ()
{   Unload ("AO-b1");
    Unload ("AO-b2");
    Unload ("Mailer2");
    Unload ("MemMgr2");
    Unload ("ThreadCreator");
    Load ("Mailer1");
    Load ("MemMgr1");
    Load ("ObjectCreator");
    Load ("AO-a1");
    Load ("AO-a2");
}
```

なお、同様な例として、Unix環境で動作するアプリケーションと、Java環境で動作するアプリケーションを考えた場合に、Unix環境とJava環境を置換する例が挙げられる。

【０１３９】４．オペレーティングシステム上での構成方法の例

オペレーティングシステム上でダウンロード機構を実現するのに、その機構をいくつかのオブジェクトに分離して実現する例について示す。

【０１４０】この例では、図１５に示すように、システムオブジェクト「Downloader」「ObjectManager」「Reflector」により、ダウンロード機構が提供される。「Downloader」は、２．３．３章で説明したロードフェーズの割当フェーズとアンロードフェーズの解放フェーズとを実現する。「ObjectManager」は、２．３．３章で説明したロードフェーズの生成フェーズとアンロードフェーズの削除フェーズとを実現する。「Reflector」は、実行環境に合わせてアプリケーションオブジェクトの初期化を行うとともに、システムオブジェクトのダウンロードの手続きの中では、ロードフェーズのシステム依存リストの更新を行う。

【０１４１】アプリケーションプログラムインターフェース「Load()」「Unload()」は、「Downloader」がサービス提供者となる。例えば、Load()が用いられると、「Downloader」が起動され、「Downloader」は、ダウンロードするオブジェクトのイメージ(コード領域及びデータ領域を含む)と、オブジェクト名などのオブジェクト情報を含むバイナリイメージとを、システム上のローカルメモリにコピーする。そして、「ObjectManager」にオブジェクトの使用するメモリ領域の初期化を依頼する。その後、「ObjectManager」により、ＩＤや実行スレッド等が生成され初期化される。そして、ダウンロードされたオブジェクトの動作を開始する。

【０１４２】５．ハードウェアの構成例

本発明が適用されるハードウェアの構成例を図１６に示す。なお、ここでは、本発明が適用されるハードウェアの構成例として、テレビジョン受信装置を例に挙げるが、当然の事ながら、本発明はその他のデータ処理装置

にも適用可能である。すなわち、本発明は、実行環境（オペレーティングシステム等）の上で実行主体（アプリケーションプログラム等）が動作するデータ処理装置に広く適用可能であり、例えば、テレビジョン受信装置以外のオーディオ・ビジュアル機器や、各種の事務機器や、汎用のコンピュータ装置等にも適応可能である。

【０１４３】本発明が適用されたデータ処理装置である図１６に示すテレビジョン受信装置は、アンテナ又はケーブル等によって放送局からの信号を受信し、当該信号に基づいて、ブラウン管又は液晶等の画像表示装置に映像を表示すると共にスピーカから音声を出力する。

【０１４４】このテレビジョン受信装置は、通常のテレビ機能を備えているだけでなく、外部からプログラムやデータを受けとることが可能となっており、図１６に示すように、バス／ＩＯブリッジ１を介してバス２に接続されたテレビ機能部３と、バス／メモリブリッジ４を介してバス２に接続されたプロセッサ５と、バス／メモリブリッジ４を介してプロセッサ５に接続されたＲＯＭ（Read Only Memory）６及びＲＡＭ（Random Access Memory）７と、バス２に接続された操作パネル８、外部記憶装置９及び通信装置１０とを備えている。

【０１４５】テレビ機能部３は、アンテナ又はケーブル等によって受信した信号に基づいて、映像や音声を再生する機能を備えている。このテレビ機能部３は、バス／ＩＯブリッジ１を介してバス２に接続されており、これにより、他の部分との信号のやり取りが可能となっている。

【０１４６】プロセッサ５は、このテレビジョン受信装置の各部の制御を行うものであり、バス／メモリブリッジ４を介してバス２に接続されている。また、プロセッサ５には、バス／メモリブリッジ４を介してＲＯＭ６及びＲＡＭ７が接続されている。ＲＯＭ６は、プロセッサ５による制御を行うためのオペレーティングシステムやアプリケーションプログラム等を記憶する。ＲＡＭ７は、プロセッサ５のワークエリアとして使われる。すなわち、プロセッサ５は、オペレーティングシステムやアプリケーションプログラムを、ＲＡＭ７をワークエリアとして使用して実行することにより、このテレビジョン受信装置を構成する各部を制御する。

【０１４７】操作パネル８は、ユーザからの操作入力を受け付けるための入力装置であり、この操作パネル８から、例えば、テレビのチャンネルやボリューム等の切り換えを指示する信号が入力される。この操作パネル８は、具体的には、各種信号を入力するための複数のボタンを備えた入力装置や、いわゆるマウスと称されるようなポインティングデバイス等からなる。この操作パネル８によって入力された信号は、バス２及びバス／メモリブリッジ４を介してプロセッサ５に入力される。そして、プロセッサ５は、操作パネル８によって入力された信号に基づいて、所定の演算処理を行って各部を制御する。

【０１４８】外部記憶装置９は、例えばハードディスク装置からなり、画像データ、制御データ、オペレーティングシステム、アプリケーションプログラム、又は外部から通信装置１０を介してダウンロードされた各種プログラム等を記録するのに使われる。また、通信装置１０は、外部との間でデータ通信を行うための入出力部であり、例えばモデムやターミナルアダプター等からなる。

【０１４９】このテレビジョン受信装置は、テレビ機能部３によって提供される通常のテレビ機能を備えているだけでなく、通信装置１０を介して、外部システムからプログラムや各種データ等を受け取ることが可能となっている。そして、このテレビジョン受信装置では、オペレーティングシステムを構成するシステムオブジェクトも、通信装置１０を介して外部システムからダウンロードすることが可能となっている。

【０１５０】また、このテレビジョン受信装置では、プロセッサ５によって、ＲＯＭ６又は外部記憶装置９に記憶されているオペレーティングシステムを実行し、このオペレーティングシステム上で、ＲＯＭ６又は外部記憶装置９に記憶されているアプリケーションプログラムを実行することにより、各部の制御を行う。すなわち、オペレーティングシステム上で、例えば、テレビ機能部３に動画像を表示するためのアプリケーションプログラムや、操作パネル８を制御するためのグラフィカル・ユーザ・インターフェース（ＧＵＩ）を実現するアプリケーションプログラムが実行される。

【０１５１】そして、このテレビジョン受信装置では、オペレーティングシステム上で動作するアプリケーションプログラムに応じて、１～４章にわたって詳細に説明したように、通信装置１０を介して外部システムからシステムオブジェクトをダウンロードして、実行するアプリケーションプログラムに応じたシステム環境を提供する。これにより、このテレビジョン装置では、様々なアプリケーションプログラムに対して常に好適に実行環境を提供することが可能となっている。

【０１５２】

【発明の効果】以上詳細に説明したように、本発明によれば、実行主体に合わせて、実行環境を構成するオブジェクトを置換することが出来るようになる。したがって、想定されるシステムサービスの機能を予め複数用意したりするようなことなく、実行主体に応じた好適な実行環境を提供することが可能となる。

【図面の簡単な説明】

【図１】システムオブジェクトダウンロードの一例を示す図である。

【図２】システムアーキテクチャの一例を示す図である。

【図３】オブジェクトの使用するメモリ領域の一例を示す図である。

【図４】ダウンロード受諾レベルとダウンロード許可レベルの一例を示す図である。

【図５】システム依存リストとサービスリストの一例を示す図である。

【図６】アンロードフェーズの前半部分である削除フェーズでの処理手順の一例を示すフローチャートである。

【図７】アンロードフェーズの後半部分である解放フェーズでの処理手順の一例を示すフローチャートである。

【図８】ロードフェーズの前半部分である割当フェーズでの処理手順の一例を示すフローチャートである。

【図９】ロードフェーズの後半部分である生成フェーズでの処理手順の一例を示すフローチャートである。

【図１０】システムオブジェクトのダウンロードによりメッセージパッシング方法を改善する例を説明するための図であり、単純なメッセージパッシング機構を持つシステムオブジェクトをダウンロードした状態を示す図である。

【図１１】システムオブジェクトのダウンロードによりメッセージパッシング方法を改善する例を説明するための図であり、優先度に応じてメッセージを処理するメッセージパッシング機構を持つシステムオブジェクトをダウンロードした状態を示す図である。

【図１２】システムオブジェクトのダウンロードによりデバッグモードを実現する例を説明するための図である。

【図１３】システムオブジェクトのダウンロードにより、２つの異なるネットワークプロトコルに適応させる例を説明するための図である。

【図１４】システムオブジェクトのダウンロードにより、アプリケーションオブジェクトの動作する動作環境全体を変更する例を説明するための図である。

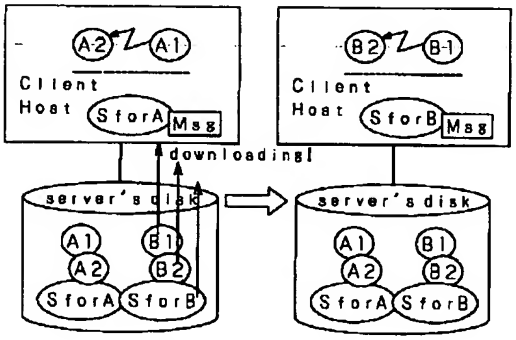
【図１５】オペレーティングシステム上でダウンロード機構を実現するのに、その機構をいくつかのオブジェクトに分離して実現する例を示すための図である。

【図１６】本発明を適用したテレビジョン受信装置の構成例を示す図である。

【符号の説明】

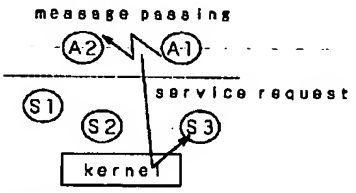
１　バス／ＩＯブリッジ、　２　バス、　３　テレビ機能部、　４　バス／メモリブリッジ、　５　プロセッサ、　６　ＲＯＭ（Read Only Memory）、　７ＲＡＭ（Random Access Memory）、　８　操作パネル、　９　外部記憶装置、１０　通信装置

【図１】



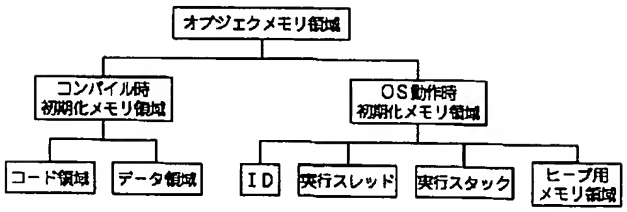
システムオブジェクトダウンロード

【図２】



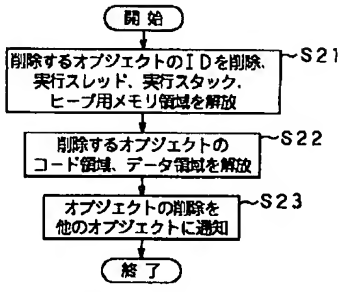
仮定するシステムアーキテクチャ

【図３】



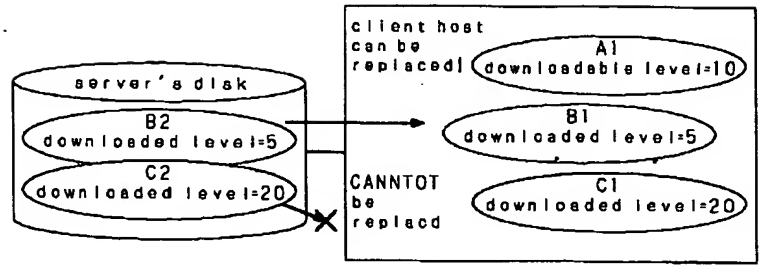
オブジェクトのメモリ領域

【図７】



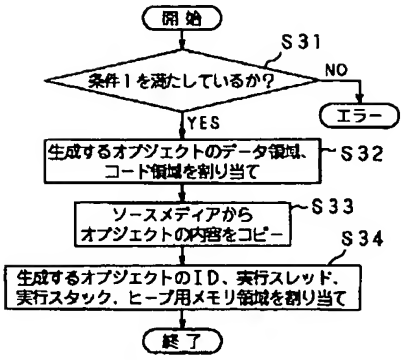
解放フェーズ

【図４】



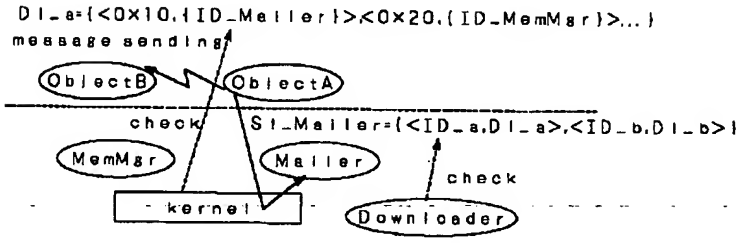
ダウンロード許可レベル、ダウンロード受託レベル

【図８】



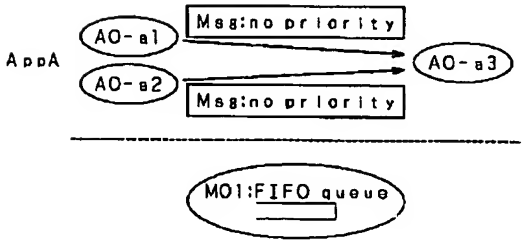
割当フェーズ

【図５】



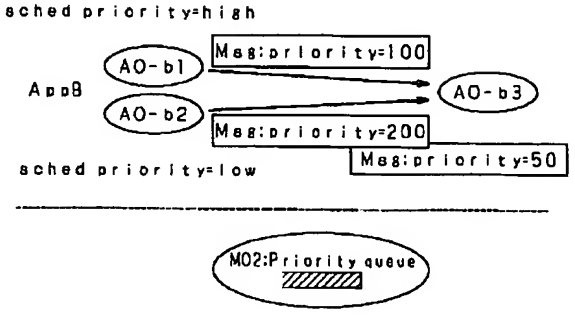
システム依存リスト、サービスリスト

【図１０】



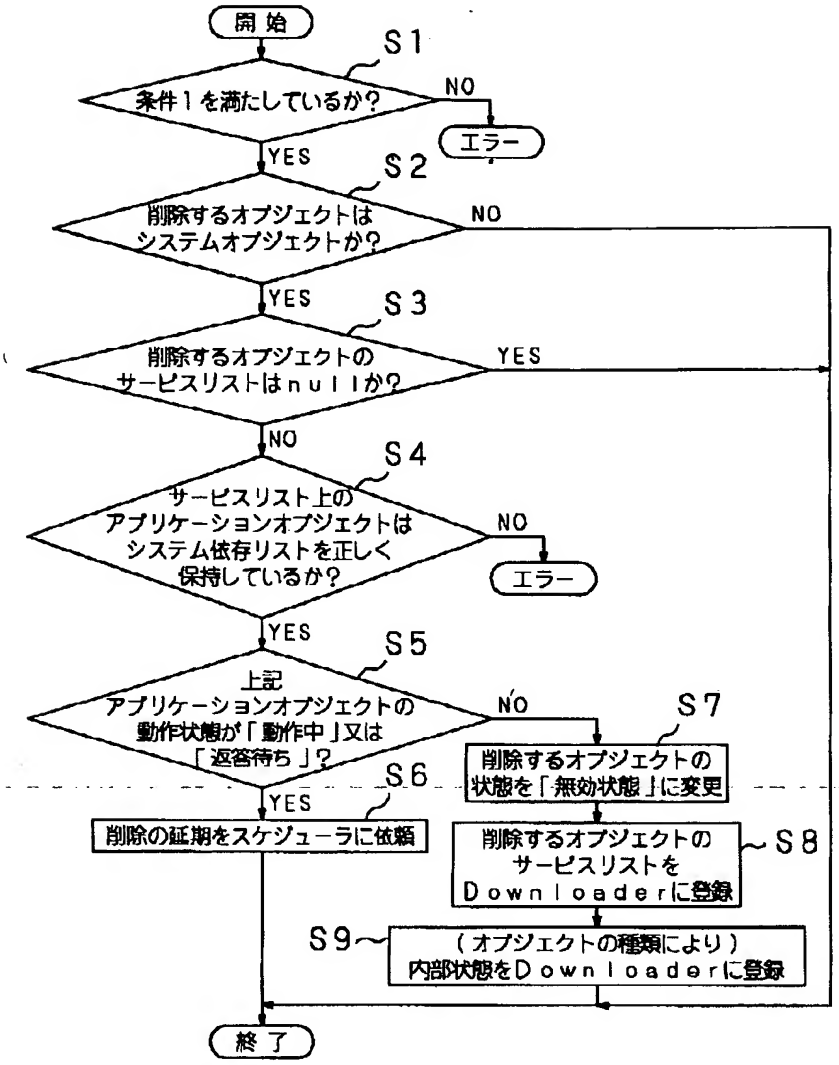
簡単なメッセージパッシング機構を用いるＡｐｐＡ

【図１１】



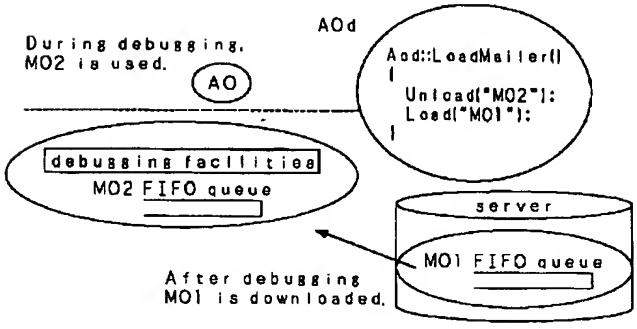
優先度付きメッセージパッシング機構を用いるＡｐｐＢ

【図６】



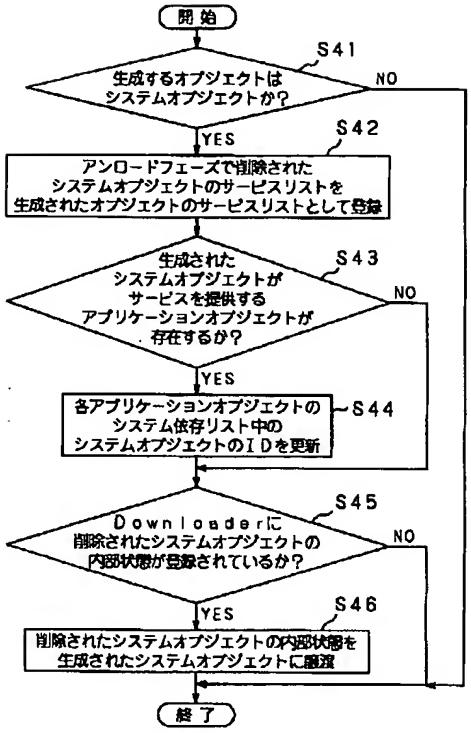
削除フェーズ

【図１２】



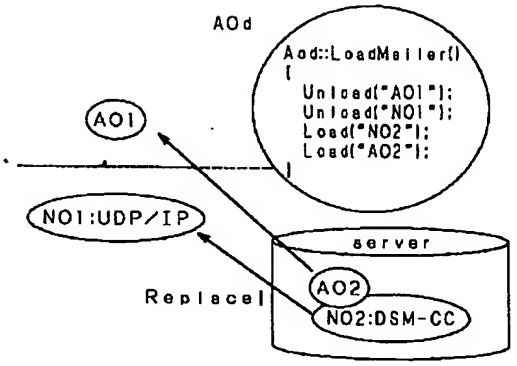
デバッグモード

【図９】



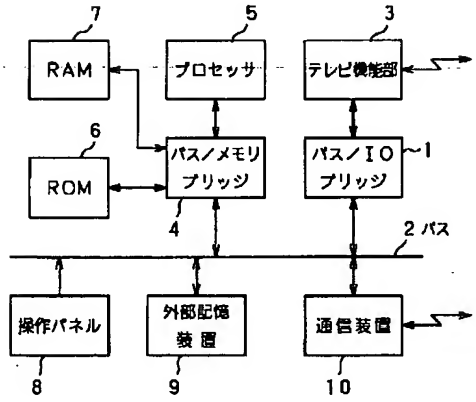
生成フェーズ

【図１３】



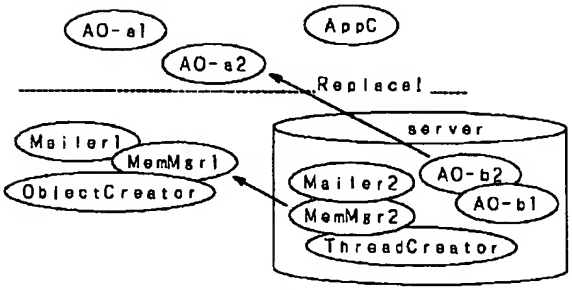
適応的ネットワークプロトコル

【図１６】



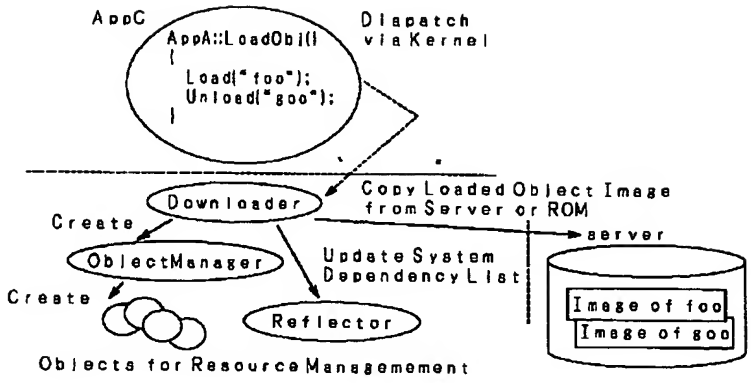
テレビジョン受信装置の概略構成

【図１４】



適応的実行環境

【図１５】



OSでのダウンロード機構の構成例